EPJ Web of Conferences **214**, 02018 (2019) https://doi.org/10.1051/epjconf/201921402018
CHEP 2018

FairRoot and ALICE O2 Multithreading Simulation

Ivana Hřivnáčová[1,*]

[1]Institut de Physique Nucléaire (IPNO), Université Paris-Sud, CNRS-IN2P3, 91406 Orsay, France

Abstract. To address the challenges of the major upgrade of the experiment, the ALICE simulations must be able to make efficient use of computing and opportunistic supercomputing resources available on the GRID. The Geant4 transport package, the performance of which has been demonstrated in a hybrid multithreading (MT) and multiprocessing (MPI) environment with up to 1/4 million threads, is therefore of a particular interest.
The O2 simulation framework is based on FairRoot, which itself is based on the Virtual Monte Carlo (VMC). The integration of multithreading into the VMC design and its impact on the Geant4 VMC were presented at CHEP 2014. Geant4 VMC multithreading and the scaling behaviour of the computing time with the number of cores have then been tested using a simplified but realistic multithreaded simulation application.
The focus was then put on the integration of multithreading in FairRoot classes as the necessary step towards multithreading in the FairRoot-based experimental frameworks. The new O2 framework is the first one of which the migration to multithreading is achieved for all actually included detectors. In this paper we present the progress with the integration of multithreading in FairRoot classes, the work for thread-safety in the O2 simulation classes and the experience with the integration of the multithreading mode in testing. We will also discuss plans for the further developments.

1 Introduction

During the LHC long shutdown LS2, the ALICE apparatus will be upgraded in order to make high precision measurements of rare probes at low pT. The ALICE Collaboration has therefore initiated a large research and development effort: the O^2 project [1]. The goal of this project is to design and build a completely new computing system capable of acquiring data for online processing at a rate increased by a factor 100 beyond the present limit.

To address the challenges of this major upgrade, the ALICE simulations must be able to make efficient use of computing and opportunistic supercomputing resources available on the GRID. The Geant4 [2] transport package, providing full support for multi-threaded simulation since version 10 is therefore of a particular interest. Its performance has been demonstrated in a hybrid multithreading and multiprocessing environment with up to 1/4 million threads.

The integration of multithreading into the Virtual Monte Carlo (VMC) design [3], lying at the core of both old and new ALICE simulation frameworks, and its impact on the Geant4 VMC [4] were presented at CHEP 2014 [5]. Geant4 VMC MT and the scaling behaviour

*e-mail: ivana@ipno.in2p3.fr

of the computing time with the number of cores have been then tested using a simplified but realistic multithreaded application, based on a Geant4 VMC standard example. Particles from parameterised *Pb-Pb* generation were transported through the full ALICE geometry including a realistic field map; hits were generated and stored for the TPC [6].

In this paper we will present the subsequent work that has followed for including the multithreading mode in the new ALICE simulation framework, O2 [7].

2 The new ALICE simulation software in O2

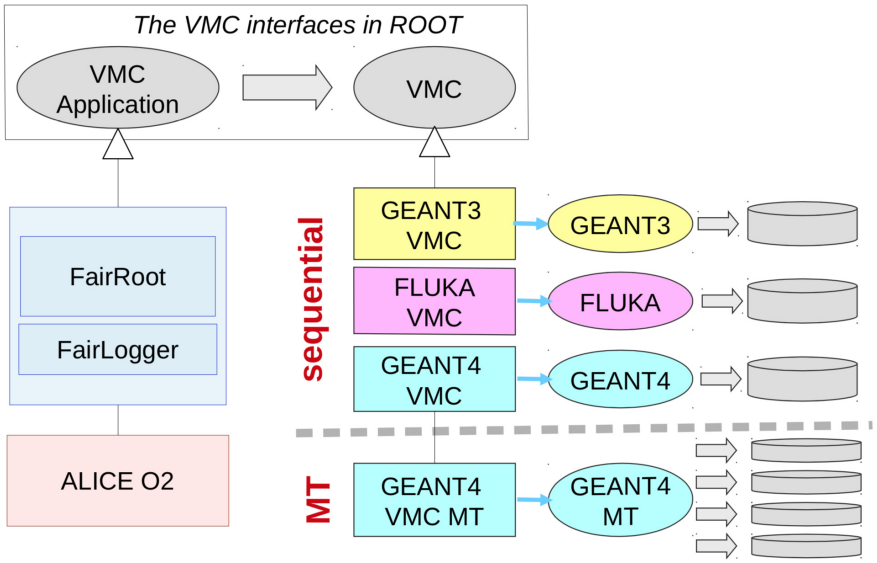

Figure 1. The VMC context in O2 framework

The new ALICE software, O2, besides general libraries and tools, is based on two other frameworks: ALFA [8] and FairRoot [9].

ALFA is the new message-queuing-based framework which is the result of a common effort of the ALICE and FAIR experiments. It was designed as a flexible, elastic system, which balances reliability and ease of development with performance using multi-processing and multithreading. It adopts a message-based approach that should support the use of the software on different hardware platforms, including heterogeneous systems, realised by the FairMQ module.

The FairRoot framework is a simulation, reconstruction and analysis framework that is based on the ROOT [10] system. It includes core services for detector simulation and offline analysis. While its development started for one FAIR experiment at GSI, it has been gradually adopted by the other FAIR and GSI experiments and beyond, including ALICE O2.

The FairRoot simulation framework is based on VMC. VMC defines an abstract layer between a detector simulation user code (MC application) and the Monte Carlo transport code (MC). In this way the user code is independent of any specific MC and can be used with different transport codes within the same simulation application. It was developed by the ALICE Offline Project and, after the complete removal of all dependencies from the

experiment specific framework, it was included in ROOT. From the three transport codes supported with VMC: GEANT 3.21 [11], Geant4 [2] and FLUKA [12], only Geant4 can run in the multithreading mode, which is why only Geant4 simulation will be discussed in this paper.

In the context of O2, the VMC application interfaces are implemented in FairRoot and some of them are further specialised in O2. Therefore the migration to the multithreading mode then concerned the packages in four hierarchy levels: the VMC category in ROOT, Geant4 VMC, FairRoot and O2. The overall picture of this context is shown in Figure 1.

3 Multithreading in Base Packages

3.1 Geant4

Geant4 offers the multithreading mode since version 10.0, released in December 2013. The Geant4 multithreading is based on a master-worker model in which one control sequence (the master) is responsible for initialising the geometry and physics, and for spawning and controlling worker threads. Workers are responsible for the simulation of one or more events. G4MTRunManager takes care of worker threads management.

Since the introduction of multithreading, Geant4 has entered the era of massive parallelism. In particular the Geant4 users have expressed strong interest in the TBB library (mainly the HEP community) and MPI integration (medical community and HPC users). Such a hybrid approach allows a simplified use of large core-count resources. Promising results from the tests run with a hybrid MPI/MT application on multiple Intel Xeon Phi cards, running up to 1000 threads without degradation of performance with respect to a perfect linear speedup, were reported for example in Ref. [13].

3.2 Geant4 VMC

The first Geant4 VMC multithreading version, 3.0, was released in 2014. The developments in this version having already been presented in [5], in this paper we just recall the main concepts relevant for the work presented here.

Geant4 VMC code was adapted for multithreading using the same approach as in Geant4 multithreading. This development consisted mainly in the replacement of the singleton objects in Geant4 VMC with singletons per thread, including the main VMC interfaces defined in ROOT: `TVirtualMC` and `TVirtualMCApplication` and fixing thread-safety issues. A new VMC package, MTRoot, implementing the ROOT output per thread with locking critical ROOT operations has been added and, finally, also the G4Root package, implementing the TGeo Geant4 navigation, was migrated.

Like for Geant4 native applications, the switching to the multithreading mode requires migration of the VMC application codes. Users need to implement new functions of `TVirtualMCApplication`, presented in [5], which are then used to clone the application and its containing objects on workers. The creation of the objects in worker threads is then triggered from the Geant4 VMC classes. Examples and more detailed instructions are available from the VMC Web site.

The multithread-specific functions introduced in ROOT with Geant4 VMC 3.0 have been revisited with the FairRoot migration. A new set of non-constant functions was added in ROOT 6.10/00, the old (constant) functions were deprecated and will be removed in the future. The actual set of these functions is presented in Table 1.

```
// Mandatory for running in the multithreading mode
virtual TVirtualMCApplication* CloneForWorker() const;
// Optional
virtual void InitOnWorker();
virtual void BeginRunOnWorker();
virtual void FinishRunOnWorker();
virtual void Merge(TVirtualMCApplication* localMCApplication);
```

Table 1. Revisited `TVirtualMCApplication` functions for multithreading.

4 Multithreading in FairRoot

The migration of FairRoot followed the recipe introduced in the VMC examples. The `FairMCApplication` class, which implements the `TVirtualMCApplication` interface, was enhanced with the new functions for multithreading presented in the previous section. Besides the mandatory `CloneForWorker()`, the `InitOnWorker()` and `FinishRunOnWorker()` functions were needed to ensure complete initialisation of the framework applications. The application initialisation was revisited so that the objects relevant to event processing are created and initialised properly on workers.

A new virtual function, the overriding of which is obligatory in the multithreading mode, was added in the `FairModule` class :

```
FairModule* CloneModule() const;
```

to ensure cloning the application scoring objects on workers. Overriding this function is, besides thread-safety, the only major item for migration to multithreading in FairRoot-based experimental simulation frameworks.

The next step, the change of all singleton classes used in simulation: `FairRunSim`, `FairStack`, `FairRootManager` and `FairLogger` to singletons per threads, was straightforward.

Finally, the `FairRootManager` class, which centrally manages the ROOT input/output operations was enhanced to automatically activate creating and writing the ROOT output files per thread when simulation runs in the multithreading mode. The implementation first followed the solution in MTRoot. Then, this class could be simplified again as most of the locking, introduced as in MTRoot, could be removed thanks to ROOT thread-safety improvements in ROOT 6.

All four examples in FairRoot/examples/simulation were migrated to multithreading. A new option to select the multithreading mode was added in the example run macros. The option is then saved in the `FairRunSim` singleton, together with the other information about the MonteCarlo setting selected by the user, and then passed to `FairMCApplication`.

The default number of threads (2) can be changed in a configuration macro using the standard Geant4 command or the environment variable G4FORCENUMBEROFTHREADS.

5 Multithreading in O2

The migration to multithreading in O2 consisted mainly of ensuring the copying of the detector and module objects in the threads and fixing thread-safety issues. All existing detectors: ITS, EMC, TRD, TOF, TPC, MCH, FIT, and passive modules: PIPE, MAG, DIPO, ABSO, actually present in the O2 framework, have been processed and enhanced with the implementation of the new `FairModule::CloneModule()` function. Also the initialisation of

sensitive volumes in detectors was revisited to get all the necessary information available in threads.

The only non-detector class requiring a modification for the multithreading mode was `o2::Data::Stack`, representing the O2 specification of the `FairMCStack` which itself provides the realisation of the `TVirtualMCStack` interface. The class required enabling and implementing of cloning the object in workers and fixing thread-safety issues.

A new option, `-isMT`, which allows activating the multithreading mode, was added to the O2 simulation application program, `o2sim`. The option is part of the set of command line options defined in the `o2::conf::SimConfig` class which can be used to configure simulation without recompiling the program or libraries.

To integrate this new simulation run mode in the standard O2 testing, the O2 build configuration files were updated to build Geant4 libraries with the `GEANT4_BUILD_MULTITHREADED` build option enabled. A new test, `o2sim_G4_mt`, has been added in the O2 standard test suites. It defines the same simulation setup (detectors, primary event generator) as in `o2sim_G3` and `o2sim_G4` tests but with multithreading enabled.

6 Outlook

At present the multithreading mode is being tested only with a small number of events in the scope of the pull request validation. A larger scale testing should follow to achieve the required code robustness. In the future we plan to improve the output from threads by adding a possibility of buffering per thread, like `G4cout` in Geant4, and also support for reading primary particles from a file in a multithreaded run.

An integration with the new parallel simulation system based on FairMQ, also presented at the CHEP 2018 conference [14], is foreseen in the near future.

7 Conclusions

In this paper we presented the achievement of the complete migration of the FairRoot framework to multithreading simulation mode available via Geant4 VMC. This development has permitted hiding most of the complexity of multithreading from the FairRoot users, the experimental frameworks, as well as minimising the necessary changes for their migration. The adapted FairRoot simulation examples can serve as tutorials for the FairRoot-based experiments and also for the standard tests.

The accomplished migration of O2 framework demonstrates the ease of the FairRoot-based framework migration. Including the multithreading mode in the standard test suite should help keep this mode running in spite of the intensive developments in both the FairRoot and O2 frameworks.

References

[1] Ananya, et al, Journal of Physics: Conference Series **513**, 012037 (2014)

[2] S Agostinelli et, Nucl. Instrum, and Methods **A506**, 250-303 (2003),
J Allison et al, IEEE Transactions on Nuclear Science **53 No. 1** 270-278 (2006),
J Allison et al, Nuclear Instruments and Methods in Physics Research **A835**, 186-225 (216)

[3] https://root.cern.ch/vmc,
I Hrivnacova et al, CoRR cs.SE/0306005 (2003)

[4] https://root.cern.ch/geant4-vmc,
I Hrivnacova, Journal of Physics: Conference Series **119** 032025 (2008)
[5] I Hrivnacova, A. Gheata, Journal of Physics: Conference Series **513**, 022014 (2014)
[6] ALICE collaboration, *Technical Design Report for the Upgrade of the Online-Offline Computing System*, CERN-LHCC-2015-006 (2015)
[7] https://github.com/AliceO2Group/AliceO2
[8] M Al -Turany et al, Journal of Physics: Conference Series **664**, 072001 (2015)
[9] M Al -Turany et al, Journal of Physics: Conference Series **396**, 022001 (2012)
[10] http://root.cern.ch
[11] R Brun et al, *GEANT3 User Guide* (CERN Data Handling Division, DD/EE/84-1, 1985)
[12] A Fasso, et al, Proc. of the MonteCarlo 2000 Conference (Lisbon, Springer Verlag Berlin) 159-164 and 955-960 (2001)
[13] A Dotti et al, Proc. of IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC, 7581867 (2015)
[14] S Wenzel, "A scalable and asynchronous detector simulation system based on ALFA", CHEP 2018 (these proceedings)